REMARKS/ARGUMENTS

This case has been carefully reviewed and analyzed in view of the Official Action dated 21 September 2004. Responsive to the rejections made by the Examiner in the Official Action, Claims 1-5, 7-22 and 24-27 have been amended and are now clearer in their respective recitations. Additionally, Claims 28-29 have been appended for prosecution. Claims 1-29 will be pending in this Application upon entry of the Amendment filed herewith.

In the Official Action, the Examiner rejected Claims 1-27, as originally presented, under 35 U.S.C. § 103(a) as being unpatentable over Fetkovich, et al. (US Patent #6,681,329; hereinafter Fetkovich) in view of Baentsch, et al. (US Patent #6,496,910; hereinafter Baentsch). To summarize broadly, the Examiner correlated features of the subject invention, as originally claimed, with the integrity checking system and method of Fetkovich. The Examiner acknowledged that Fetkovich does not show the formation of tables in memory and the subsequent use of those tables for storing selected addresses. The Examiner concluded, however, that inserting tables in memory and storing addresses in tables is well known in the art and relied on Baentsch for an exemplary disclosure of such.

Prior to discussing in any further detail the prior art relied upon by the Examiner in making the rejections, it is believed beneficial to first briefly describe Applicant's invention in light of the amended Claims, the Specification and the

Drawing. The invention of the subject Patent Application provides means for verifying the integrity of a software application during the execution of the program, or "on-the-fly". Through the inventive system and method, addresses of relocatable program code are held in tables in memory so as to be available to a security application or module upon demand. Moreover, the subject invention maintains the information in memory so that temporally costly computations may be avoided during program execution. Though widely applicable, an exemplary application for which the present invention provides beneficial features is a Digital Versatile Disk (DVD) player, particularly one implemented entirely in software and executed on a standard computing platform. In such systems, security measures may be applied to data in relocatable modules and translated addresses must then be computed if any of the modules is displaced to an alternate memory location. The security measures may detect and prevent unauthorized access by a third party and may be continuously operable while the protected software is being executed. By retaining relocation data in memory, integrity of the software may be verified in real time without delays in the execution of certain program steps which would result in erratic or undesirable software behavior.

The subject invention may implement a method for determining the integrity of an application program by first allocating one or more segments of computer memory for storing at least a data portion of the application program. The allocated data portion is used, in part, to maintain one or more tables of

selected addresses of relocatable computer code of the program. For this to be achieved, the invention implements certain steps during the software build process.

According to certain aspects of the invention, building the application program utilizes a linker, which is operable to associate addresses across relocatable modules and further operable to output relocation data stored within the relocatable modules. The linker is used to link one or more relocatable object modules with one or more libraries and other object modules, each having relocation data stored therein, to form an intermediate executable module. During this linking operation, addresses of portions of the libraries and other modules that are linked to the intermediate executable module are selected. The applicable addresses are determined by examining the relocation data output by the linker during the linking phase for, say, a jump or function call from one portion of the application program to a memory location of a relocatable portion thereof linked thereto from the previously-mentioned libraries and other modules. The selected addresses are stored in the tables maintained in the data portion previously allocated.

During the build process, a default address of a selected subprogram of the intermediate executable module is stored in the data portion. The libraries and other object modules are then loaded into memory to produce the executable application program. It is important to note that the relocation data is retained in

tables in the data portion after the application program has been loaded. As will be discussed further below, prior art systems and methods do not maintain the relocation data once the application program is ready to be executed.

At any time after the program begins execution, a reference address associated with the previously selected subprogram is compared with the default address thereof. In some embodiments, substitute addresses for the addresses stored in the tables are computed if the reference address differs from the default address. A security application is executed using addresses computed from the reference address and either the selected addresses in the tables or the substitute addresses to determine the integrity of the application program.

In certain embodiments of the subject invention, the security application performs decryption on previously encrypted program code. In other embodiments, the integrity is validated by computing a checksum. As the relocation data is readily available in the data segment of memory, the security application can perform either of these integrity verification procedures in real time while the application is executing. This is particularly beneficial for dynamic systems allowing code relocation even while other portions of the same program are in an execution phase or are otherwise in a committed operation, such as in a pipeline.

In making the rejections, the Examiner asserted that tables in which addresses are stored are widely used in the art. While this is in a broad sense true,

the invention of the subject Patent Application creates a table in memory to hold the relocation data that is otherwise discarded from memory after the linking and loading phases of the application program build process. The subject invention thus provides a specific, useful and non-obvious service over the prior art, i.e., supplying relocation data to a security application upon demand.

In contradistinction with the invention of the subject Patent Application, the integrity checking method of Fetkovich is centered on the use of a digital signature, which, once computed, is independent of the memory location of the relocatable module. The method of Fetkovich does not require the addresses of the relocatable code once the digital signature has been determined. For that reason, Fetkovich does not disclose the use of tables, as was acknowledged by the Examiner. In fact, the addition of tables to Fetkovich would be at a minimum superfluous, and possibly even performance-hampering, since valuable memory would be occupied by information that the method of Fetkovich does not require. Thus, it is respectfully submitted that adding tables to hold relocation data to Fetkovich would require a change in the principle of operation of the invention disclosed therein in order to utilize the modification. That being so, it is submitted, respectfully, that an ordinarily skilled artisan would not be motivated by examination of Fetkovich to modify the system and method thereof by the use of tables to store relocation addresses subsequent to the conclusion of the build

process and, as such, the invention of the subject Patent Application is not made obvious by the reference.

Arguendo, even if the use of tables were applied to Fetkovich, acquisition of the relocation data by "providing a linker... operable to output relocation data stored in ... relocatable modules", "selecting addresses ... by examining during said linking step the relocation data output by said linker" and "storing said selected addresses in said tables", as recited by the amended Claims of the subject Patent Application, is neither explicitly disclosed nor even suggested by the reference. It is believed that this formation of the tables for the purposes of the verification of integrity of software composed of relocatable modules is both novel and non-obvious, in that it involves unique modifications to the software build process to prepare the addresses subsequently provided to a security application.

The Examiner used the apparatus and method of Baentsch as an exemplary application of the use of tables during linking and loading of software. Whereas, Baentsch makes use of a relocation table during the linking and loading process, this table is deleted at the conclusion thereof in favor of using the memory space for other purposes (column 5, lines 53-55; column 6, lines 59-65). Thus, applying Baentsch to Fetkovich does not produce a system and method for integrity checking of relocatable program code where "said tables [are] retained in said at least one data portion during said program execution" in preparation for "executing a security application or module to verify the integrity of the

application program at addresses determined from [a] reference address and ... selected addresses in said tables", such as is now recited by the amended Claims of the subject Patent Application.

Independent Claim 1, as now amended, recites more clearly the inventive method as including the steps of "allocating ... at least a data portion of the application program", "inserting tables in said data portion", "providing a linker ... operable to output relocation data", "selecting addresses of portions ... linked to [an] intermediate executable module by examining during said linking step said relocation data output by said linker", "storing said selected addresses in said table", "said tables being retained in said ...data portion during ... application execution" and "executing a security application or module to verify the integrity of the application program at addresses determined from [a] reference address and said selected addresses in said tables". As previously discussed, the combination of these steps are not shown or suggested by any of the references cited by the Examiner in making the rejections of the Claims. Moreover, a combination of references fails to show the unique combination of method steps for the purposes of verifying the validity of relocatable program code during execution of the program from which the relocatable program code is formed. Thus, it is believed that Independent Claim 1 is both novel and non-obvious over the cited references and is in condition for allowance. That being so, and as Claims 2-10 are

ultimately dependent from Claim 1, it is also believed that the Dependent Claims 2-10 are allowable for at least the same reasons for which Claim 1 is allowable.

Currently amended Independent Claim 11 recites a computer system having a central processing unit and memory and, among other elements, "a program code translator... including ... a program code linker, said linker operable to ... output relocation data stored in ... relocatable modules", "at least one application program built by said program translator..., said application program including at least one data segment having formed therein at least one address table, said address table having stored therein selected addresses of program code from one or more relocatable object modules as linked together by said linker, said selected addresses determined from said relocation data stored in said one or more relocatable object modules as output by said linker", and "[a] security application operable to access said table while said application program is executing". These elements are not shown nor are they suggested by any of the references cited by the Examiner. Furthermore, a combination of references cited by the Examiner fails to show the unique combination of elements recited by Claim 11 for the purposes of on-the-fly integrity checking of relocatable code. Thus, Independent Claim 11 is believed to be in condition for allowance. Moreover, as Claims 12-16 and new Claims 28-29 are ultimately dependent from Independent Claim 11, the Dependent Claims are believed to be allowable for at least the same reasons for which Claim 11 is allowable.

Independent Claim 17, as now amended, recites a computer readable medium having computer-executable instructions, which when executed on a computer system perform the inventive method of the subject Patent Application. The method steps recited by the Claim closely correspond to those recited in Claim 1. Thus, for the reasons discussed above with reference to Claim 1, Claim 17 is believed to be allowable, and Claims 18-19, which are dependent from Claim 17, are believed to be allowable for at least the same reasons for which the Independent Claim 17 is allowable.

Currently amended Claim 20 recites the method of the subject invention as including the steps of "inserting tables into pre-allocated memory segments residing in [a] data space", "examining relocation data for selected addresses when ... pre-compiled object files are linked by a program code linker with one or more libraries and other object files and loaded into the memory, said program code linker operable to output relocation data ..., said relocation data examination being of said relocation data ... upon output by said program code linker", "storing said selected addresses in said tables", "said tables ... being retained in the data space until the application program is unloaded from the memory" and "performing a checksum ... at addresses determined from [a] reference address and said selected addresses in said tables". The method steps recited are not shown or suggested by any of the references cited by the Examiner nor are they shown by a combination of those references. It is therefore believed that Independent Claim 20 is in

condition for allowance and that Claims 21-24, which are ultimately dependent from Claim 20, are believed to be allowable for at least those reasons for which Claim 20 is allowable.

Claim 25, as currently amended, recites a computer readable medium having computer-executable instructions, which when executed on a computer system performs the inventive method steps of "allocating ... at least a data portion of the application program", "inserting tables in said data portion", "linking via a program code linker one or more relocatable object modules with one or more libraries and other object modules ..., said program code linker operable to output ... relocation data", "selecting addresses ... by examining during said linking step said relocation data output by said program code linker", "storing said selected addresses in said tables", "executing the application program ..., said tables being retained in said data portion during said application program execution" and "executing a security application or module to verify the integrity of the application program at addresses determined from said reference address and said selected addresses in said tables". By reasoning similar to that already given, it is believed that Claim 25 is allowable and that the Dependent Claims based thereon, i.e., Claims 26-27, are allowable for at least the same reasons for which the Independent Claim is allowable.

The remaining references cited by the Examiner, but were not used in the rejections, have been reviewed and are believed to be further remote from the

subject Patent Application than the references used by the Examiner when patentable considerations are taken into account.

In view of the foregoing amendments and remarks, Applicant believes that the subject Patent Application is in condition for allowance and such action is respectfully requested.

Respectfully submitted,
For: ROSENBERG, KLEIN & LEE

Morton J. Rosenberg
Registration #26,049

Dated: 12/3/04